Bridging the Open Web and APIs: Alternative Social Media Alongside the Corporate Web

JACK JAMIESON, University of Toronto, Canada

DRAFT July 17, 2019. NOT FOR CIRCULATION OR CITATION.

What compromises and challenges occur when alternative social media rely on corporate platforms with which they have philosophical and material differences? This question is explored in an empirical study of the IndieWeb, a community of personal websites with social networking features, including the ability to syndicate content to and from corporate social media platforms using their Application Programming Interfaces (APIs). I study the development history of Bridgy, a tool for this syndication, and identify three main challenges arising from its relationship with Facebook: Difficulty translating between the open Web and APIs; occasional ambiguity in Facebook's presentation of privacy settings; and ongoing precarity due to API updates. Bridgy's responses to these problems illuminate power relationships between platforms and third-party developers. This raises considerations for building alternative social media tools and for internet researchers who rely on platform APIs.

1 INTRODUCTION

Recent concerns about the dominance of social media giants highlight a need to consider alternative ways of engaging with the social Web [16, 19, 29]. In large part, objections to corporate social media have advocated opting-out, as demonstrated by hashtag campaigns like *#deletefacebook* and efforts to build alternative social media (ASM) platforms such as Diaspora and Mastodon. Given the pervasiveness of Facebook and other corporate platforms, persuading users to leave poses a significant challenge. Some technologists have pursued a more moderate approach, attempting to reconfigure rather than abandon relationships with corporate social media.

This article investigates frictions and compromises that arise when ASM operates alongside and even within the infrastructures to which they claim to provide an alternative. Specifically, I discuss IndieWeb, a "people focused alternative to the 'corporate web'" in which individuals create and operate personal Web sites that function as their primary online identity [24]. IndieWeb has developed standards, software, and practices that support peer-to-peer communication among personal Web sites while also maintaining connections with large platforms. Rather than avoiding corporate social media, content from personal Web sites is syndicated to Facebook, Twitter, and other platforms, and then comments, likes, and other responses are retrieved back.

Most ASM reflect their creators' principles above economic concerns [16] and IndieWeb is no exception. I leverage scholarship about values and design to investigate how IndieWeb's creators attempt to articulate their principles in the design of technical artifacts [e.g. 15, 38]. This 'translation' from values to artifact is complicated by IndieWeb's entanglement with corporate platforms through its syndication flow. One of IndieWeb's founders has remarked, "We still use the silos [corporate platforms] as a distribution mechanism. We will use the tools of our competitors, of our enemies, to further our own causes" [9]. Unsurprisingly, there have been cases where those "enemies" restricted types or quantities of data transmission in ways that compromised IndieWeb's goals.

The simultaneously dependent and antagonistic relationship between IndieWeb and corporate platforms highlights the heterogeneous environment in which ASM operate. I investigate this relationship through a study of Bridgy, a tool to syndicate content and responses between personal IndieWeb sites and platform APIs. This paper describes Bridgy's relationship with Facebook's API

Author's address: Jack Jamieson, jack.jamieson@mail.utoronto.ca, University of Toronto, Toronto, Canada.

from 2014 through 2018, addressing the following questions: (1) How might IndieWeb's goals be challenged by its reliance on corporate APIs? (2) If problems arise, how are they addressed?

Whereas Helmond highlighted how "social media platforms are enacting their programmability to reweave the web for social media" [20], Bridgy attempts to reconfigure the relationship between Web and social media in a different direction. This paper highlights challenges of interoperating between the open Web and platforms, as well as the precarity of relying on platform APIs. This will contribute to knowledge about how to build systems that reflect their creators' principles when growing alongside the corporate Web.

1.1 IndieWeb

According to its website, "the IndieWeb is a community of individual personal websites, connected by simple standards, based on the principles of owning your domain, using it as your primary identity, to publish on your own site (optionally syndicate elsewhere), and own your data" [25]. Three reasons are presented for using the IndieWeb instead of corporate platforms: "Your content is yours"; "You are better connected"; and "You are in control" [24]. IndieWeb's co-founder Tantek Çelik has asserted that these three commitments all stem from one goal: self-empowerment [10]. Individuals are to be empowered by both owning and controlling their Web content, rather than relying wholly on platforms for hosting. However, individuals are also to be empowered by connecting to "all services, not just one" [24]. IndieWeb community members commonly post first to their personal Web site, and then syndicate to a variety of other platforms.

IndieWeb's community has produced several standards for adding social features to personal websites. Two of the most salient for this paper are "Microformats 2," a semantic markup language with which one can add machine-readable context to HTML, and "Webmention," a standard for notifying when one webpage links to another. In combination, these standards allow a wide variety of interactions to be communicated between individual websites, such as replies, likes, shares and other actions familiar on social media platforms.

As described in the Webmention spec [34], if Bob uses his website to write a reply to Alice, he can send a Webmention with the following content:

```
source=http://www.bob.example/post-by-bob
target=http://www.alice.example/post-by-alice
```

"Source" indicates a URL for the page that is sending a Webmention and "target" indicates the recipient. If Alice wants, she may set her site up to display Bob's reply, in which case her site will retrieve the content from the source URL, and parse its Microformats 2 markup to determine the type of reply. For example, Bob may add markup indicating the post is a "like", in which case Alice's website may choose to display something like the following below her post content: "Bob liked this post."

As demonstrated by this example, Alice has lots of options for what to with the Webmention she receives. This is characteristic of IndieWeb's structure of interoperating yet independent websites. Rather than a single project or platform, IndieWeb consists of a plurality of standards, content management systems, Web and mobile apps, design patterns, and other components. Standards like Webmention, therefore, are designed to be generalizable, and can work across websites with significantly different software and designs. Individual control over one's website is a core aspect of this system.

1.2 Syndicating to platforms with Bridgy

One of IndieWeb's defining features is its emphasis on maintaining connections to platforms through an approach called POSSE (Publish on Own Site, Syndicate Elsewhere). Although IndieWeb's premise

is that people use their own websites as their main online presence, they do not avoid corporate social media altogether. IndieWeb users usually syndicate content from their personal Web sites to Facebook, Twitter, and other platforms, and then retrieve comments, likes, and other responses back to the original post. Çelik has described two advantages to this approach: 1) It allows one to keep in touch with the many people who use popular social media platforms, 2) by aggregating "interactions from Twitter, independent websites, and Google Plus, all in one place" one "has an experience on his site that's better than any silo" [9].

The most popular tool for syndicating content between IndieWeb sites and platforms is Bridgy. As described on its website, Bridgy "pulls comments, likes, and reshares on social networks back to your Web site. You can also use it to post to social networks - or comment, like, reshare, or even RSVP - from your own web site" [7]. Like many IndieWeb tools, Bridgy is provided for free and is funded out of pocket by its creator. Bridgy works by translating between popular platform APIs and IndieWeb sites that are built with standards like Webmention, Microformats 2, and conventional Web standards including HTTP, URLs, and HTML.

In contrast to the generalizable IndieWeb standards described earlier, APIs vary across platforms, each allowing access to different types of data and requiring distinct methods and identifiers to facilitate that access. Bridgy can connect to APIs for Twitter, Google Plus, Instagram, Flickr, and GitHub, though its features for each platform vary according to constraints of individual APIs. For example, Bridgy's documentation [7] explains that Twitter's search API is "best effort only" so "every now and then [Bridgy] may not be able to find one of your replies". Significantly, Bridgy used to support Facebook, but this feature was shut down in August 2018 when, in response to the Cambridge Analytica scandal, Facebook dramatically restricted their API, removing capabilities that were necessary for Bridgy's function.

2 LITERATURE REVIEW

2.1 Values and design

By attempting to empower individuals through the creation of software and protocols, IndieWeb exhibits a commitment to the idea that social change can be nurtured through design work. This demonstrates a close alignment with the commitments of values and design research for building technologies that enhance "human well-being, human dignity, justice, welfare, and human rights" [15].

An extensive body of scholarship in science and technology studies and information studies has made it clear that values can be instantiated in sociotechnical systems [e.g. 43]. One of the pressing concerns of values and design research is explaining how this occurs. Most values and design research approaches this question with an interactional position, affirming that "whereas the features or properties that people design into technologies more readily support certain values and hinder others, the technology's actual use depends on the goals of the people interacting with it" [15]. This perspective builds upon affordance theory [17, 32], which asserts that environments and technologies support and constrain (though do not determine) potential directions for action. By recognizing that the actual use of a technology is shaped not only by affordances but also by the needs and perspectives of users, the interactional perspective of values and design research recognizes interpretive flexibility in both the design and use of technologies [38]. Technologies may be designed to support particular values, but the realization of those values depends on the context into which they are deployed. Further, Humphreys [23] observes that the stabilization that occurs during a technology's maturation is temporary, and so possibilities for intepretive flexibility can resurface when the context surrounding a technology changes. Therefore, in addition to studying

the role of values in the initial design of technologies, there is a need to address the ongoing work involved in repairing and maintaining those values.

This is particularly clear in the realm of Web technologies, which involve a multitude of explicit relationships to other systems, and since the advent of "Web 2.0" increasingly follow a model of constant iterative development [33]. The ability to supplement and repurpose platforms has been one of the great sources of promise in analyses of Web 2.0 [37, 39]. Nonetheless, developers who build upon platforms work with an awareness that the foundations upon which they rely may be changed or deprecated suddenly, requiring unexpected work to maintain or repair core functions. Addressing the maintenance and repair of values in technology reframes those values as "contingent and ongoing accomplishments" [22]. This draws attention to variety of external pressures that can constrain values over the duration of an artifact.

2.2 Infrastructures and platforms

Highlighting contingencies around values and design shifts attention to broader infrastructures into which ASM are embedded. Web technologies operate in heterogenous systems of protocols, servers, software, hardware, people, policies, and other resources. This leads to "a baffling network of relationships producing significant outcomes that no single actor seems particularly able to foresee" [36]. As such, designers attempting to articulate values through infrastructural systems must contend with ongoing unpredictability. Infrastructure studies and platform studies are two closely related approaches that have been used to investigate the systems that constitute the web. In this section, I describe how both approaches inform my analysis.

The Web as conceived by inventor Tim Berners-Lee is distinctly infrastructural [35]. It is based on open standards such as HTML, HTTP, and URLs, which have been formed incrementally through consensus among multiple stakeholders. It nurtures interoperability among diverse components and resources and is embedded into daily life. When operating smoothly, infrastructural systems tend to retreat into the background. It is typically upon breakdown that infrastructures reveal themselves (Star, 1999). Because IndieWeb's relationships with corporate platforms and broader Web infrastructures involve efforts to re-purpose them in sometimes unexpected ways, breakdowns are common. As a result, IndieWeb provides a useful site for performing an 'infrastructural inversion' (Star, 1999) that foregrounds the systems upon which it is built. The purpose of this inversion is not to focus only on material parts of a system, but instead to uncover tacit labour [36] and thereby draw attention to "the political, ethical, and social choices that have been made throughout its development" [4].

One of the major differences between the open Web and social media platforms is in their attitude toward openness. Berners-Lee's original conception of the Web as a network of linked documents was open in that (1) a document could be accessed by anyone who knew its URL, and (2) anyone could put a document online without having to first seek permission. In contrast, he has argued that much content on Web platforms is enclosed in 'walled gardens' or 'silos' [2]. This means that rather than being able to access content simply by knowing its URL, one has create an account with the platform hosting that content. Creating an account with most platforms requires agreeing to a Terms of Service (TOS) agreement, challenging the 'permissionless' quality of the Web. Although platforms have generally lowered technical and expertise barriers for participation online, they reserve a right to revoke access to uses and users they deem illegitimate [e.g. 18]. Platforms have nurtured new forms of communication among people from all over the world, but the openness provided by these tools is contingent upon centralized gatekeepers.

Montfort and Bogost [31] and Helmond [20] have advocated using a material-technical perspective of platforms to investigate "the connection between technical specifics and culture" [3]. One of the key technical features of Web platforms is their programmability through APIs. This article

focuses on this computational dimension to examine developers' efforts to interoperate between platforms and the open Web.

Plantin et al. assert that "digital technologies have made possible a 'platformization' of infrastructure and an 'infrastructuralization' of platforms" [35]. They explain that infrastructure studies emphasizes "ubiquity, reliability, invisibility, gateways, and breakdown" while platform studies highlights "programmability, affordances and constraints, connection of heterogeneous actors, and accessibility of data and logic through application programming interfaces (APIs)" [35]. To account for the convergence of infrastructures and platforms, they call for a combination of both perspectives. This is particularly significant for studying an artifact like Bridgy, that explicitly joins the open web to platform APIs.

3 METHOD

As with most IndieWeb projects, Bridgy is open source and hosted on GitHub, a platform for software code and other version-controlled repositories. GitHub repositories include a detailed history of revisions to source code and other documents, making it possible to identify how a project has changed over time. Individual changes are published as "commits" and discussion threads about new features and bugs are archived as "issues." Issues provide an apt entry-point for investigating Bridgy's connections to other systems. First, they highlight breakdowns, since many of issues literally describe errors, bugs, and other problems with Bridgy. Second, they present traces of the work that was conducted to resolve these breakdowns. In many cases, these discussions reference specific commits in which an issue was addressed or resolved. As a result, it can be possible to observe deliberation in developer discussions about issues, and then identify ways in which issues were addressed through code.

I downloaded copies of Bridgy's issues for analysis.[1] Between the earliest item (January 4, 2014) and the time of downloading (March 5, 2018), there were 799 issues. I then performed a keyword search to get a general sense of how many issues pertained to each platform, with the following results: Twitter (N=316), Facebook (N=278), Google (N=165), Instagram (N=99), Flickr (N=50), Blogger (N=32).[2] This is not an exact measure because the presence of a keyword does not guarantee that the issue actually pertains to that platform, and some keywords are ambiguous ("Google" can refer to a variety of products, and Bridgy only has an API relationship with Google Plus). Nonetheless, this preliminary query contributes a rough sketch of where Bridgy has had the most issues in relation to corporate platforms.

I selected issues involving Facebook for further analysis because (1) they represent a significant proportion of Bridgy's total issues, and (2) Facebook's large scope means it demonstrates many different types of interactions and potential problems. In an interview, Bridgy's creator indicated that Facebook is a good representation of Bridgy's relationship with corporate Web platforms (silos) more generally: "Any feature in any silo, Facebook has it too. And then they also have ten or a hundred thousand features that silo doesn't. There aren't many concerns that you don't see in Facebook."

A first pass reading of the 278 issues that included the term "Facebook" revealed that 147 of them described some sort of problem or feature request related to Facebook's API. I then used open coding to organize the issues into categories that described types of problems that were most prominent. Where possible, I followed links to code commits that were referenced in the issue

[1] The script used for downloading GitHub issues is available at (removed for peer review).

[2] At the time of this study, Bridgy had recently added compatibility for syndicating to and from GitHub. This feature was not included in this study because it was brand new and lacked a significant number of issues for analysis.

discussions, which assisted in identifying how Bridgy's developers responded to different types of challenges.

While studying GitHub data, I was mindful for potential pitfalls presented by Kalliamvakou et al. [27]. While most of these pitfalls refer to large-scale quantitative studies, and so are not relevant to this study, I will address two pitfalls that needed to be considered. First, like many projects, Bridgy is not contained wholly within a single repository, instead relying on pieces from other repositories to perform some of its functions. Studying a single repository without considering its dependencies can lead to an incomplete understanding. A benefit of treating Bridgy's issues as the entry for this study is that the issues posted to the main Bridgy repository often reference commits to its dependencies. Therefore, it was possible to start with Bridgy and expand outward to its dependencies, avoiding a potential blind spot.

Second, GitHub projects almost always involve development and discussion occurring outside of the GitHub platform. Accordingly, this study is supplemented by ongoing participant-observation of IndieWeb's developer community and a semi-structured interview with Bridgy's creator and lead developer, Ryan Barrett. This interview served to verify my assessment of the technical challenges and approaches as discovered through analysis of its GitHub repository, as well as to develop a richer understanding of Barrett's motivations and experiences.

4 FINDINGS

The issues reported on Bridgy's GitHub reference a wide variety of topics. However, three types of problems emerged as the most substantial and recurring causes of breakdowns. Below, I summarize these problems as well as ways in which developers responded to them.

4.1 Mapping between URLs and API IDs

When syndicating between personal Web sites and Facebook, Bridgy spans a threshold between the open Web and platform APIs. The most prominent challenge in Bridgy's development history has been translating between different ways of addressing an object across this boundary. Objects on the open Web are addressed using URLs, such as *http://facebook.com/{user-id}/{object-id}*, whereas the same object could be identified within Facebook's API using an ID in a format such as *{user-id}_{object-id}*.

In some instances, it is straightforward to translate between these formats. In the example above, one can see how it could be possible to map between these identifiers using *{user-id}* and *{object-id}*. However, this mapping is often quite difficult for a variety of reasons. The first reason is that Facebook IDs come in a several different formats, represented in Table 1 based on a comment in Bridgy's source code. Bridgy's developers struggled to predict a rationale for which format is required in different cases, and Bridgy's code resorts to a sequence of guessing and trial and error to find the correct format.

Mapping between URLs and API IDs became more difficult in 2014, when Facebook released version 2.0 of its API. This update limited the amount of data third-party developers could access. In 2018, When Facebook CEO Mark Zuckerberg testified before the U.S. House of Representatives Committee on Energy and Commerce about Cambridge Analytica's collection of Facebook users' data, he asserted that this update "makes it so a developer today can't do what Kogan [the developer who shared data from his app with Cambridge Analytica] did years ago" [45]. One of the changes was the introduction of app-scoped user IDs, which mean that each third-party app is given a different ID for the same user. This improves Facebook users' privacy and security because it frustrates efforts to combine data collected by multiple apps. However, this also complicates Bridgy's efforts to map between a URL on Facebook.com and the corresponding object in Facebook's API.

Table 1. Types of formats for Facebook IDs

Format	Example
"Simple number, usually a user or post"	12
"Two numbers with underscore, usually POST_COMMENT or USER_POST"	12_34
"Three numbers with underscores, USER_POST_COMMENT"	12_34_56
"Three numbers with colons, USER:POST:SHARD"	12:34:56
"Four numbers with colons/underscore, USER:POST:SHARD_COMMENT"	12:34:56_56

This led a notable reduction in Bridgy's capabilities. Prior to this update, Bridgy could be used to like a Facebook post from one's own website, without visiting Facebook directly. To do so, one would write a post to their website that linked to a Facebook post and included Microformats 2 markup to designate the post as a "like". For example, a post might include the following HTML:

```
Alice liked <a href='{URL}' 'class='u-like-of'>a post on Facebook</a>.
```

Bridgy could interpret the post's HTML and, if the URL points to a post on Facebook.com, could find the matching object in Facebook's API and then post a "like" on the user's behalf. The reason this is no longer possible is that URLs on Facebook.com refer to users by either their username or a global user ID, and neither of these identifiers can be mapped to the app-scoped ID Bridgy must use when navigating Facebook's API. As a result, Bridgy's developers removed support for this feature.

Although Bridgy's ability to syndicate likes from a personal website to a Facebook account was removed, in most cases Bridgy has been successful at mapping between URLs and API IDs, albeit with considerable effort. From Facebook's perspective, IDs are intended to be taken as opaque objects, and Bridgy's attempts to decode them are not supported. Facebook support has advised Bridgy's developers, "Please treat IDs as unique strings, they are not meant to be broken down and used" [13]. However, it is only when broken down and used that these IDs can be translated in URLs. To accomplish this, Bridgy relies on heuristics and trial and error, as explained by its lead developer:

There's no consistent way, either through the API or through an algorithm you implement yourself, to say, 'Here is a Facebook post, what is its permalink on Facebook.com?' We have to guess at that with a surprising number of heuristics. Not ideal. So that mapping back and forth between the Web and data inside Facebook has been the single biggest question. Bridging the open Web with Facebook's API necessitates reconciling different meanings and expectations on either side. Boundary objects are one way to address differences in the meaning of Facebook content across this divide. Boundary objects acts as "a means of translation" across multiple communities of practice [5], meeting local requirements while also maintaining a common identity. This allows different groups to work together in the absence of consensus by "tacking" back and forth between an object's general form, which is meaningful across communities, and specific forms of the object tailored to meet local needs [40]. Bridgy and Facebook share a general understanding of IDs as a means of referring to objects, but differ significantly in their local expectations of how IDs should be used. As a result, Bridgy uses heuristics, trial and error, and similar methods to fit the shared

general understanding of IDs with its local requirements. This is an example of articulation work, "the continuous efforts required in order to bring together discontinuous elements" [41]. Creating systems that interoperate requires significant investments of time and energy on relatively invisible work.

4.2 Mapping privacy from front-end to back-end

There are two main mechanisms for users to control who accesses their data on Facebook. First, they may use Facebook's privacy settings to specify an audience that can view their posts and other information on Facebook.com or in the Facebook app. Options include "Public", "Friends", "Friends except…", "Specific friends", and "Only me." Second, whenever a third-party app such as Bridgy wants to access Facebook data, users must grant it permission. For example, to access a user's posts and photos, a third-party app will display a notice that the app will receive "your timeline posts and photos" — If the user agrees, the app will be granted the *user_posts* and *user_photos* permissions and will be able to access that data in Facebook's API.

These two privacy mechanisms work independently. Notably, there is no way to grant a third-party developer access to public photos, but not to photos intended for friends only. As a result, it has been up to third-party developers to manage their treatment of posts with different audiences. This is significant for Bridgy because it processes only public data. When Bridgy retrieves data from Facebook to one's personal website, it publishes a publicly visible copy of that interaction on Bridgy's website. This is necessary to assign each interaction a URL, which is required to send a valid Webmention.

When accessing content in Facebook's API, Bridgy checks its privacy status and ignores any content not marked explicitly as 'public.' Barrett commented on this process:

> That is non-trivial to determine for a given object in the Facebook API, is it public? I mean usually, 90% of the time, it's straightforward. Another 9% it takes some work, but you can figure it out, again depending on the type. it's like 1% or maybe 0.1% where you actually can't tell. You look at the inheritance chain, you look at a bunch of other stuff, and you just don't know. If you go look at the UI [User Interface] in Facebook, you can usually tell. But programmatically you can't. And so when that happens, I have to err on the side of not doing anything with it.

The most striking example of this difficulty occurs with some photos uploaded to Facebook. In Facebook's API, photos themselves do not possess a privacy status. Instead, each photo is part of a parent post and/or album, which contains a privacy field indicating the intended audience.[3] Therefore, determining the privacy status for a photo in Facebook's requires one to check this parent.

When someone posts multiple photos in a short period of time, Facebook creates a parent post representing all of them as a group, even if the individual photos have distinct privacy settings. In these cases, the parent post may have a privacy status of "CUSTOM". In a Github issue, Barrett explained that Facebook's documentation "[does not] say anything about what CUSTOM with no details means."[4] As a result, there are rare cases when it is not possible for Bridgy to determine the privacy status of an object in the API, even if it would be apparent through Facebook's user interface. Archived discussions between Bridgy's developers and Facebook support suggest that this problem is related to Bridgy's unconventional way of navigating Facebook's API, highlighting how efforts to repurpose platforms can surface hidden features of their systems [12].

[3]The privacy field for an object in Facebook's specifies a privacy setting among these options: EVERYONE, ALL_FRIENDS, FRIENDS_OF_FRIENDS, SELF, CUSTOM.

[4]https://github.com/snarfed/bridgy/issues/611#issuecomment-174315265

To avoid accidentally publishing non-public data, Bridgy's approach is to ignore any data that is not explicitly marked as public. This has led to cases where Bridgy fails to process content that is intended to be public if the privacy setting in Facebook's API is unclear. This has been interpreted by multiple users as a bug on Bridgy's part, since the data is marked as public in Facebook's user interface and yet was ignored by Bridgy.

This example helps position the importance of privacy in Bridgy's design. IndieWeb is generally less concerned with privacy than some other alternative social media, as evidenced by practices such as syndicating content to corporate platforms and the fact that IndieWeb sites are almost always publicly accessible and indexable by search-engines. However, even though Bridgy does little to enhance individuals' privacy, it is designed to avoid infringing upon existing privacy expectations. Shilton et al.'s [38] sociotechnical dimensions of values provide axes for describing the place of privacy in Bridgy's design: Salience (peripheral to central), intention (accidental to purposive), and enactment (potential to performed). In many use-cases, privacy is a potential rather than performed value in Bridgy, since the software usually deals with public data. When there is a potential infringement, however, it becomes clear that privacy has a high salience and intention, since it supersedes Bridgy's proper functioning. The result is a value dam [30] where privacy is so central to Bridgy's lead developer that he opposes conflicting designs, even when it limits the software's perceived efficacy.

4.3 Precarity and API updates

APIs can change quickly and unpredictably. While this study was underway, Facebook issued substantial API updates to improve its security and privacy in light of the Cambridge Analytica scandal. These updates, removed the ability for third-party apps to publish content to one's Facebook account (Archibong, 2018), which meant that Bridgy could no longer syndicate from one's website to Facebook. A subsequent update introduced restrictions that limited Bridgy's ability to send comments and likes from Facebook back to one's website. As a result, Bridgy dropped support for Facebook altogether [1].

Until 2018, the precarity of Bridgy's relationship with Facebook was somewhat subtle. For the most part, Bridgy's developers had been able to maintain functionality with Facebook's API, albeit with considerable expertise and effort. By studying Bridgy's development history, this study was able to illuminate the labour of responding to API updates, and particularly the decision-making to preserve values such as privacy. As reliance on third-party APIs has become a prominent feature of software development, the impact of API updates have been investigated in several studies [11, 21, 44]. Nonetheless, excepting those who experience bugs or participate in development processes, the precarity of Bridgy and other software that rely on third-party APIs is often hidden.

During our interview, Barrett was quick to assert that problems he encountered with Facebook's API were not a result of malice. Rather, "in many ways what Bridgy is doing is not at all what Facebook expects the average Facebook app to do." Further, he asserted that an app like Bridgy is simply too small to be a concern for Facebook, citing his past work experiences as a senior engineer at Google: "I have seen some it at the inside of Google. For things that aren't security breaches—you know for apps that are just doing funny things that may or may not be against your TOS [terms of service]—if they're small enough you don't care." The development history of Bridgy indicates that Facebook and similar platforms are not actively attempting to prevent Bridgy's style of syndication. In fact, in the past Facebook has solicited work on a similar tool from software developer and open Web advocate Dave Winer [26]. In this case, it is unlikely that anyone at Facebook was specifically motivated to restrict services like Bridgy. Instead, Facebook's attitude toward Bridgy could be characterized as indifference.

This indifference creates opportunities for experimentation and innovation, but as demonstrated by Facebook's dramatic API updates in 2018, also cultivates substantial risk. Another IndieWeb developer who created a commercial service with similar syndication features as Bridgy has written that reliance on platform APIs had been an obstacle, especially from a business perspective, "Back when I was working on Known, investors would ask about the supplier risk of being so heavily dependent on third party APIs to provide a lot of the core value. They were right" [42].

One of the goals of this study was to consider how skilled developers might have agency to contest platform features with which they disagree, rather than simply opting out or proposing wholly separate alternatives. Although Bridgy was, for a time, successful at navigating its relationship with Facebook, the pressure of the Cambridge Analytica scandal prompted a set of restrictions in Facebook's API that Bridgy could not accommodate. Third-party developers who use platform APIs can build upon platforms with greater strategic agency than users who take the system as-is, but still must work within the affordances of tools provided by platform operators.

5 DISCUSSION AND CONCLUSION

Investigating challenges in Bridgy's development from 2014 through 2018 presents a novel perspective of how Facebook's API impacted third-party developers during this period. The examples highlighted in this paper demonstrate key challenges for alternative social media, particularly those that attempt to maintain combatibility with existing systems. This is not only important for cases like Bridgy that interoperate directly with platform APIs, but also for any system that attempts to work across internet infrastructures.

Attempts to work across the open Web and platforms must confront distinct logics of identifying and addressing resources. In Bridgy's case, these logics could usually be managed through heuristics and other forms of articulation work. Such challenges cannot be resolved outright but instead require ongoing maintenance and repair. Additionally, difficulty determining the privacy status of some objects led to an ethical dilemma, although this was relatively rare. The result was to preserve users' privacy with a design that compromised Bridgy's functionality. And ultimately, although most updates could be navigated, major changes to Facebook's API in 2018 led Bridgy's developers to drop support for Facebook altogether.

Beyond the scope of alternative social media, Bridgy's responses to these challenges hold lessons for any who use platform APIs, such as computational researchers studying social media. Internet researchers have warned that Facebook's API restrictions present significant obstacles for critical and public-interest research about platforms [8]. Freelon [14] proposed that researchers should incorporate Web scraping into their data collection, and highlighted technical, legal, and ethical considerations for doing so. Bridgy's translation between IndieWeb sites and platform APIs relies on scraping and then parsing HTML that has been structured for machine-readability. The challenges posed during this translation highlight additional factors for researchers considering Web scraping. App-scoped IDs for API objects may not be mappable to URLs used to identify the same data on the Web, which may hinder analysis between Web scraped and API-based datasets. Conversely, because the privacy status of some Facebook posts may be presented unclearly in the API, Web scraped data may provide a clearer sense of users' privacy expectations, as long as these are clearly presented on a platform's front-end. However, researchers should still be mindful of users' lack of clarity of these settings even when visible in a platform's interface [6, 28]. Finally, since platform HTML is unlikely to be structured with machine-readability in mind, Web scraping can require significant planning and setup. Coupled with the frequency at which platforms can update their data structures or front-end layouts, this poses a special challenge for longitudinal studies that incorporate Web scraping.

Given the extent of platformization, many systems and tools must co-exist with corporate platforms to some extent. Builders of alternative social media propose different social and technical arrangements than corporate platforms. While this leads to challenges, builders of alternative social media push at the boundaries of existing systems, revealing points at which these systems can be adapted. As technologists and researchers ponder ways to improve the ethical conditions of future social media, understanding how existing systems may accept or resist specific forms of change can help us plan for achievable and sustainable projects.

5.1 Limitations and future work

The methods used in this study are contingent upon the thoroughness in which issues are documented in Bridgy's GitHub repository, which may limit their utility in other contexts. Additionally, this study's reliance on GitHub for its primary data source limits its ability to identify phenomena that are only observable elsewhere. Future work will investigate interoperation among a broader set of related systems.

ACKNOWLEDGMENTS

This research was supported by the Social Sciences and Humanities Research Council of Canada.

REFERENCES

[1] Ryan Barrett. 2018. RIP Facebook for Bridgy. https://snarfed.org/2018-08-03_rip-facebook-for-bridgy
[2] Tim Berners-Lee. 2010. Long Live the Web. *Scientific American* 303, 6 (2010), 80–85. http://www.nature.com/scientificamerican/journal/v303/n6/full/scientificamerican1210-80.html
[3] Ian Bogost and Nick Montfort. 2009. Platform Studies: Frequently Questioned Answers. *Digital Arts and Culture* (2009), 7.
[4] Geoffrey C. Bowker, Karen Baker, Florence Millerand, and David Ribes. 2009. Toward Information Infrastructure Studies: Ways of Knowing in a Networked Environment. In *International Handbook of Internet Research*, Jeremy Hunsinger, Lisbeth Klastrup, and Matthew Allen (Eds.). Springer Netherlands, Dordrecht, 97–117. http://link.springer.com/10.1007/978-1-4020-9789-8_5
[5] Geoffrey C. Bowker and Susan Leigh Star. 2000. *Sorting Things out: Classification and Its Consequences*. The MIT Press, Cambridge, Mass.
[6] danah boyd and Eszter Hargittai. 2010. Facebook Privacy Settings: Who Cares? *First Monday* 15, 8 (July 2010). https://doi.org/10.5210/fm.v15i8.3086
[7] Bridgy. 2018. About - Bridgy. https://brid.gy/about
[8] Axel Bruns. 2018. Facebook Shuts the Gate after the Horse Has Bolted, and Hurts Real Research in the Process. https://policyreview.info/articles/news/facebook-shuts-gate-after-horse-has-bolted-and-hurts-real-research-process/786
[9] Tantek Çelik. 2014. Why We Need the IndieWeb. https://www.youtube.com/watch?v=HNmKO7Gr4TE
[10] Tantek Çelik. 2016. State of the IndieWeb. https://www.youtube.com/watch?v=7zTolqW_I2g
[11] Tiago Espinha, Andy Zaidman, and Hans-Gerhard Gross. 2014. Web API Growing Pains: Stories from Client Developers and Their Code. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014*. IEEE, Antwerp, Belgium, 84–93. https://doi.org/10.1109/CSMR-WCRE.2014.6747228
[12] Facebook for Developers. [n. d.]. Post Not Returning with Correct Privacy. https://developers.facebook.com/bugs/983622558349347/
[13] Facebook for Developers. [n. d.]. Unusually Formatted Comment Ids (with Colons) Returned by /Me/Posts. https://developers.facebook.com/bugs/786903278061433/
[14] Deen Freelon. 2018. Computational Research in the Post-API Age. *Political Communication* 35, 4 (Oct. 2018), 665–668. https://doi.org/10.1080/10584609.2018.1477506
[15] Batya Friedman and Peter H. Kahn. 2003. Human Values, Ethics, and Design. In *Handbook of Human-Computer Interaction*, J Jacko and A Sears (Eds.). Lawrence Erlbaum Associates, Mahwah, NJ, 1177–1201.
[16] Robert W. Gehl. 2015. The Case for Alternative Social Media. *Social Media + Society* 1, 2 (2015), 1–12. https://doi.org/10.1177/2056305115604338
[17] James J. Gibson. 1986. The Theory of Affordances. In *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 127–143.

[18] Oliver L. Haimson and Anna Lauren Hoffmann. 2016. Constructing and Enforcing "Authentic" Identity Online: Facebook, Real Names, and Non-Normative Identities. *First Monday* 21, 6 (June 2016). http://firstmonday.org/ojs/index.php/fm/article/view/6791

[19] Harry Halpin. 2018. Decentralizing the Social Web: Can Blockchains Solve Ten Years of Standardization Failure of the Social Web?. In *INSCI'2018- 5th International Conference 'Internet Science'*. St. Petersburg, Russia, 16.

[20] Anne Helmond. July-December 2015. The Platformization of the Web: Making Web Data Platform Ready. *Social Media + Society* 1, 2 (July-December 2015), 1–11. https://doi.org/10.1177/2056305115603080

[21] Andre Hora, Romain Robbes, Nicolas Anquetil, Anne Etien, Stephane Ducasse, and Marco Tulio Valente. 2015. How Do Developers React to API Evolution? The Pharo Ecosystem Case. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference On*. IEEE, 251–260. https://doi.org/10.1109/ICSM.2015.7332471

[22] Lara Houston, Steven J. Jackson, Daniela K. Rosner, Syed Ishtiaque Ahmed, Meg Young, and Laewoo Kang. 2016. Values in Repair. In *CHI '16 Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM Press, 1403–1414. https://doi.org/10.1145/2858036.2858470

[23] Lee Humphreys. 2005. Reframing Social Groups, Closure, and Stabilization in the Social Construction of Technology. *Social Epistemology* 19, 2-3 (Jan. 2005), 231–253. https://doi.org/10.1080/02691720500145449

[24] IndieWeb.org. 2018. Home. https://indieweb.org/

[25] IndieWeb.org. 2018. IndieWeb. https://indieweb.org/IndieWeb

[26] Mathew Ingram. 2014. Don't like Facebook Owning and Controlling Your Content? Use Tools That Support the Open Web. https://gigaom.com/2014/09/03/dont-like-facebook-owning-and-controlling-your-content-use-tools-that-support-the-open-web/

[27] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An In-Depth Study of the Promises and Perils of Mining GitHub. *Empirical Software Engineering* 21, 5 (Oct. 2016), 2035–2071. https://doi.org/10.1007/s10664-015-9393-5

[28] Yabing Liu and Krishna P Gummadi. November 02 - 04, 2011. Analyzing Facebook Privacy Settings: User Expectations vs. Reality. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. Berlin, Germany, 61–70.

[29] Geert Lovink and Miriam Rasch (Eds.). 2013. *Unlike Us Reader: Social Media Monopolies and Their Alternative*. Number 8 in Inc Reader. Inst. of Network Cultures, Amsterdam. OCLC: 840018305.

[30] Jessica K. Miller, Batya Friedman, Gavin Jancke, and Brian Gill. 2007. Value Tensions in Design: The Value Sensitive Design, Development, and Appropriation of a Corporation's Groupware System. In *Proceedings of the 2007 International ACM Conference on Supporting Group Work (GROUP '07)*. ACM, New York, NY, USA, 281–290. https://doi.org/10.1145/1316624.1316668

[31] Nick Montfort and Ian Bogost. 2009. *Racing the Beam: The Atari Video Computer System*. MIT Press, Cambridge. https://mitpress.mit.edu/books/racing-beam

[32] Donald A. Norman. 1999. Affordance, Conventions, and Design. *interactions* 6, 3 (1999), 38–43. http://dl.acm.org/citation.cfm?id=301168

[33] Tim O'Reilly. 2005. What Is Web 2.0. http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=1

[34] Aaron Parecki. 2017. Webmention. https://www.w3.org/TR/webmention/

[35] Jean-Christophe Plantin, Carl Lagoze, Paul N. Edwards, and Christian Sandvig. 2016. Infrastructure Studies Meet Platform Studies in the Age of Google and Facebook. *new media & society* (2016), 1–18. http://nms.sagepub.com/content/early/2016/08/02/1461444816661553.abstract

[36] Christian Sandvig. 2013. The Internet as Infrastructure. In *The Oxford Handbook of Internet Studies*, William H. Dutton (Ed.). Oxford University Press, Oxford.

[37] Mirko Tobias Schäfer. 2011. *Bastard Culture!: How User Participation Transforms Cultural Production*. Amsterdam University Press, Amsterdam.

[38] K. Shilton. 2013. Values Levers: Building Ethics into Design. *Science, Technology & Human Values* 38, 3 (May 2013), 374–397. https://doi.org/10.1177/0162243912436985

[39] Clay Shirky. 2008. *Here Comes Everybody: The Power of Organizing without Organizations*. Penguin, New York.

[40] Susan Leigh Star. 2010. This Is Not a Boundary Object: Reflections on the Origin of a Concept. *Science, Technology & Human Values* 35, 5 (Sept. 2010), 601–617. https://doi.org/10.1177/0162243910377624

[41] Lucy Suchman. 1996. Supporting Articulation Work. In *Computerization and Controversy*, Rob Kling (Ed.). Academic Press, San Diego, CA, 407–423. http://books.google.com/books/about/Computerization_and_Controversy.html?id=9wlN9eOomacC

[42] Ben Werdmüller. 2018. I'm Done with Syndication. Let's Help People Be Themselves on the Web. https://werd.io/2018/im-done-with-syndication-lets-help-people-be-themselves-on

[43] Langdon Winner. 1980. Do Artifacts Have Politics? In *Daedalus*. http://www.jstor.org/stable/10.2307/20024652

[44] Laerte Xavier, Aline Brito, Andre Hora, and Marco Tulio Valente. 2017. Historical and Impact Analysis of API Breaking Changes: A Large-Scale Study. In *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference On*. IEEE, Klagenfurt, Austria, 138–147. https://doi.org/10.1109/SANER.2017.7884616

[45] Mark Zuckerberg. 2018. Hearing Before the United States House of Representatives Committee on Energy and Commerce. https://docs.house.gov/meetings/IF/IF02/20180508/108260/HHRG-115-IF02-Wstate-BarrettG-20180508.pdf